DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Proposal for Master′s Thesis in Automotive Software Engineering

Hardware-in-the-Loop Setup for Machine-Learning-Based Schedulability Analysis on a Model Car using Embedded Hardware, NeuroRobotics Platform and Neuromorphic Hardware

Octavio Ivan Delgadillo Ruiz

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Proposal for Master's Thesis in Automotive Software Engineering

Hardware-in-the-Loop Setup for Machine-Learning-Based Schedulability Analysis on a Model Car using Embedded Hardware, NeuroRobotics Platform and Neuromorphic Hardware

Hardware-in-the-Loop Setup für Machine-Learning-Basierte Schedulability-Analyse auf einem Modellauto mit Embedded Hardware, NeuroRobotics Platform und Neuromorphe Hardware

Author:	Octavio Ivan Delgadillo Ruiz
Supervisor:	Dr. Prof. Uwe Baumgarten
Advisor:	M. Sc. Bernhard Blieninger
Registration Date:	15.06.2020

Contents

1 Introduction

One of the biggest trends in the automotive industry is the building of increasingly autonomous vehicles that free users from some tasks, and so increase their comfort and safety. This should be combined with other relevant trends, such as the protection of the environment and the efficient usage of resources, in order to be sustainable. Therefore, as computers are performing more and more tasks in modern cars, it is sensible to ensure their efficient utilization, but this has not been the case so far.

Modern cars include dozens of small computers, a number that is likely to increase if we consider past trends [Vip+14; Vip18]. These computers, commonly known as electronic control units (ECUs), are embedded devices that often must meet certain constraints to ensure the safety of the tasks performed, such as real-time capabilities. This is one reason why a lot of devices are integrated to serve a specific purpose. But when so many devices are part of a system, many problems arise, for example, inefficient usage of the devices (since many tasks are only executed in specific situations), reduced fault tolerance in case a system fails, and increased weight and cost of the vehicle due to the high number of ECUs and cables [Vip18]. Hence, the automotive industry seeks alternative approaches that help solve these issues, while prioritizing vehicle safety.

One such approach is the concept of ECU consolidation, which aims to run tasks on a few powerful, multi-purpose ECUs, instead of using lots of single-purpose ECUs. Such consolidation brings other challenges with it, such as higher computation and safety requirements, especially in case of a software or hardware failure (as many tasks could be blocked by a single failure) and for some safety-critical tasks, where redundancy is required [Mun+17]. To cope with these issues, previous projects at the institute, such as KIA4SM and MaLSAMi, have introduced the concept of dynamic task migration, which aims to solve these problems by enabling the execution of tasks on various devices, depending on the state of the system. This permits a re-distribution of the tasks onto different devices. To plan the task distribution, schedulability analysis is performed to ensure all tasks can be executed and meet their real-time constraints. Researchers in these projects have explored machine learning algorithms as an interesting alternative to traditional mathematical approaches for performing schedulability analysis. The mathematical approaches, such as the real-time analysis algorithm proposed by Buttazzo in [But11], often require recurrent calculations that consider the dependencies between tasks and shared resources; this means that in spite of providing exact calculations, they are often too complex and computation intensive for large task sets. In contrast, the machine-learning-based alternative has shown the potential of being a fast and acceptable approximation of the feasibility of the distribution of the tasks.

While the results delivered by previous work seem promising, more evidence is necessary to determine whether machine learning can be used for schedulability analysis on different platforms. Therefore, testing the approach with different pieces of hardware and machine learning algorithms is important to gather such evidence. First evidence has been collected with the hardware setup used in previous work, such as the Hardware-in-the-Loop (HIL) setup used in KIA4SM.

The purpose of this master's thesis is to build a Hardware-in-the-Loop setup to evaluate a system using machine learning and an adaptation of the task migration concept to improve reliability, load distribution and system utilization for automotive architectures. While the idea of task migration is the base for this setup, its full implementation is beyond the scope of the thesis and the concept will be explored in the form of the planning and deploying of a new task distribution, unlike the one in KIA4SM. This work shall implement the setup by using the NeuroRobotics Platform (NRP), embedded hardware running FreeRTOS, a simple model car and a SpiNNaker neuromorphic board for the machine learning algorithm. While the HIL setup implemented in this work is based on the one used in KIA4SM, it shall use completely different tools, including the introduction of the neuromorphic architecture for the machine learning tasks. Ideally, this should serve to prove that machine-learning-based schedulability analysis can be adapted to new platforms and perform optimally. To demonstrate this, the system will be tested using time analysis under artificially created critical situations.

2 Related Work

Research projects KIA4SM and MaLSAMi have explored through initial experiments the possibility of migrating tasks running on a device to another in a real-time capable system (for example, ECUs in a vehicle) under certain conditions. In these works, the migration was divided into two main stages: The first is the migration planning, which determines the hardware that tasks will be migrated to, should the original hardware not be able to fulfill its duty (for example, if there is a failure in that hardware or if the real-time constraints or deadlines would be violated). The second is the execution of the migration, which ensures that corresponding tasks can be migrated from the source device to the target device while keeping their current state. The scope of this thesis only covers the migration planning; hence, the second stage is not relevant for this work.

MaLSAMi analyzed migration planning, with researchers performing schedulability analysis based on machine learning (specifically, on neural networks). Machine learning was picked over traditional mathematical approaches, because those can become too complex for big task sets and often lead to a pessimistic calculation of the system utilization. By predicting the feasibility of a task set using machine learning algorithms, we can obtain potentially faster, but less precise, results, as demonstrated by previous theses by Taieb, Utz and Blieninger [Tai19; Utz19; Bli19]. The predictions provided by the machine-learning approach indicate whether a task set is 100% schedulable or not, but they are not completely safe, since false positives can occur. For this purpose, false positives could be punished heavily and a percentage of success could be predicted for a task distribution in future states at any given time, but this idea has not been investigated yet. This makes this a potentially powerful solution to enable the execution at run-time of the real-time capable migration planning.

For this purpose, a neuromorphic platform using spiking neural networks is an interesting approach, due to the benefits its architecture provides in terms of speed and energy-efficiency when compared to a traditional architecture. These advantages are present in neuromorphic architectures because they work in a similar way to brains and are therefore well suited for tasks where the brain normally outperforms traditional computers, such as learning. The SpiNNaker board offers such an architecture, which can be connected with relative ease to a robot body using the NeuroRobotics Platform (NRP). MaLSAMi also produced a data set for training the neural networks used for the schedulability analysis. This data set was generated on real hardware running on GenodeOS, which executed different task sets formed by some dummy tasks. The execution times of the tasks and task sets were measured using End-to-End Measurement, which helps provide a better estimation of the system load. The relevance of these training data remains to be proven for different system configurations.

Additionally, KIA4SM showcased the possibility of building a Hardware-in-the-Loop setup for testing the task migration approach on a specific hardware. In this setup, the SpeedDreams simulator offers the virtual environment for some autonomous driving tasks, and the boards running GenodeOS execute the tasks. This serves as a basis for the setup to be implemented in this thesis as explained in the next section.

Although all these projects have achieved research-relevant results in their segments, more proof is needed to show the adaptability of the machine-learning-based migration planning on different hardware and software configurations. Therefore, the work in this thesis shall allow for testing the capabilities of the developments in combination with neuromorphic hardware and the NRP.

3 Description of the Approach

As explained above, in this master's thesis, I aim to develop a Hardware-in-the-Loop setup for a model car or robot, using the NeuroRobotics Platform, embedded hardware and a neuromorphic platform, which will be running a machine learning algorithm for migration planning. This would allow for a new task distribution in a given failure scenario for one of the devices in the system. The NRP will provide the virtual environment for the car and the embedded hardware shall mimic a car to show its capabilities in a real situation.

The proposed Hardware-in-the-Loop setup consists of a virtual domain and a real, embedded domain, which interact with each other through the exchange of important information, such as sensor data and control commands. The virtual domain includes the NRP scenario, along with the ROS topics or services and the Gazebo model for the robot; this stage emulates the sensors and actuators as interfaces with the environment. The embedded domain refers to the model car architecture, running on embedded devices with FreeRTOS; these devices are responsible for running the autonomous driving tasks, such as pathfinding, reading signals from the sensors, generating signals for motor control, and some dummy tasks to keep system utilization high. Additionally, the machine learning algorithm running on a SpiNNaker neuromorphic board allows for a correct distribution of the tasks between the systems to mitigate critical situations, such as the failure of a board.

The NRP scenario will present a pathfinding task for the robot in a labyrinth. This will be further extended by allowing for the connection with the real embedded hardware mimicking the car. This connection shall enable the exchange of the relevant sensor and control signals and will be implemented by ROS topics or services. The model car architecture consists of various embedded boards running on FreeRTOS, which will execute the afore mentioned tasks with certain real-time constraints. The path planning task shall be implemented, along with the reading of sensors and generation of control signals. Dummy tasks, on the other hand will be provided by COBRA framework, along with their execution times.

Since the scope of this thesis should not exceed a total of 6 months, some parts of the machine learning approach shall be taken from previous works, especially the training data and some ideas for the neural network to be implemented. The training data that has been previously generated in those projects shall be reutilized as much as possible, and for this purpose the dummy tasks shall be generated with the COBRA framework and shall resemble the ones utilized in MaLSAMi. Since the machine learning algorithms have not been implemented on the SpiNNaker board yet, a feed-forward net will be the initial approach, because it can be transformed to run on the neuromorphic board.

Therefore, to achieve the goal of the thesis, which is showing that the machine-learning-based approach for schedulability analysis is adaptable to new platforms and performs decently, following stages must be accomplished to build the proposed Hardware-in-the-Loop setup:

1. Building an NRP scenario for a robot with an application for path planning
2. Integration of the model car in the NeuroRobotics Platform as the real robot. This way, the HIL setup can run the tasks in the embedded devices
3. Porting of the application in 1. to the real model car, so it can be executed in the embedded board, and development of additional dummy applications to ensure system utilization will be high to critical
4. Creating an application failure scenario using the hybrid simulation setup (NRP + model car)
5. Mitigation of the application failure / recovery of the system using task re-deployment based on migration planning and machine learning tools from other works
6. Timing evaluation of the results

To be achieved, each stage should be subdivided in tasks, as described next:

1. The NRP scenario is cloned from an existing scenario in the NRP, which places a virtual robot in a labyrinth. Taking this as a starting point, the path planning algorithm shall be implemented, so that the robot detects the distance to the walls in the labyrinth and searches for an exit. Here, the important tasks to be considered are ensuring that the robot configuration allows for the correct navigation of the environment and a first version of the path planning algorithm in Python.
2. The virtual robot in the NRP shall communicate with the model car by means of ROS topics or services. These shall allow the embedded devices in the model car to execute the path planning tasks and send control signals to the virtual robot and to receive the emulated sensor inputs. The tasks to be considered in this

stage are establishing the communication of both parts of the setup, the definition of the type of interfaces to be used (topics or services) and their implementation on the NRP simulation and on the embedded hardware.

3. The application developed in 1. shall be executed on real devices running on FreeRTOS. For this purpose, the application must be rewritten to C++ code, extending with the usage of the ROS topics or services developed in 2. for reading the sensors and controlling the robot. The dummy tasks shall be generated using COBRA framework and basic FreeRTOS tasks, to resemble the tasks used in MaLSAMi. Using COBRA framework, we can also generate task sets and estimate Worst-Case-Execution-Times, which will be used later as input for the machine learning algorithm.

4. A failure scenario shall be created to make the system under the original task distribution fail in fulfilling all real-time constraints for the task set. The failure scenario must ensure that the system is brought to its limits, so following are possible scenarios: a) suddenly shutting down one of the embedded boards, b) blocking a task on a board (for example, by means of a deadlock), c) firing new tasks that force a board's task set to be unschedulable. Relevant tasks in this stage are the decision and implementation of the strategy for causing the failure, as well as ensuring that the system is brought to a critical state.

5. Based on previous work, a machine learning approach for schedulability analysis shall be implemented on the SpiNNaker board, Therefore, the first task here is to create a feed-forward net that can be transformed to the neuromorphic board for this use case. Then, the connection between the boards and the migration planning algorithm should be set. Afterwards, the planned distribution shall be achieved by redeploying the tasks onto the devices that are still running. Here, it shall be ensured that a feasible task distribution is selected.

6. With hardware or software tools, the time between the failure and the deployment of the new task configuration shall be measured. For this, time consumed in intermediate steps (such as the task distribution planning and the deployment) might be measured using End-to-End measurement. The results of these measurements shall be compared then to ones from previous works, such as those by Utz [Utz19] and Blieninger [Bli19].

This task definition permits the fulfillment of the goal of the master's thesis in the planned time, as detailed in the time plan below.

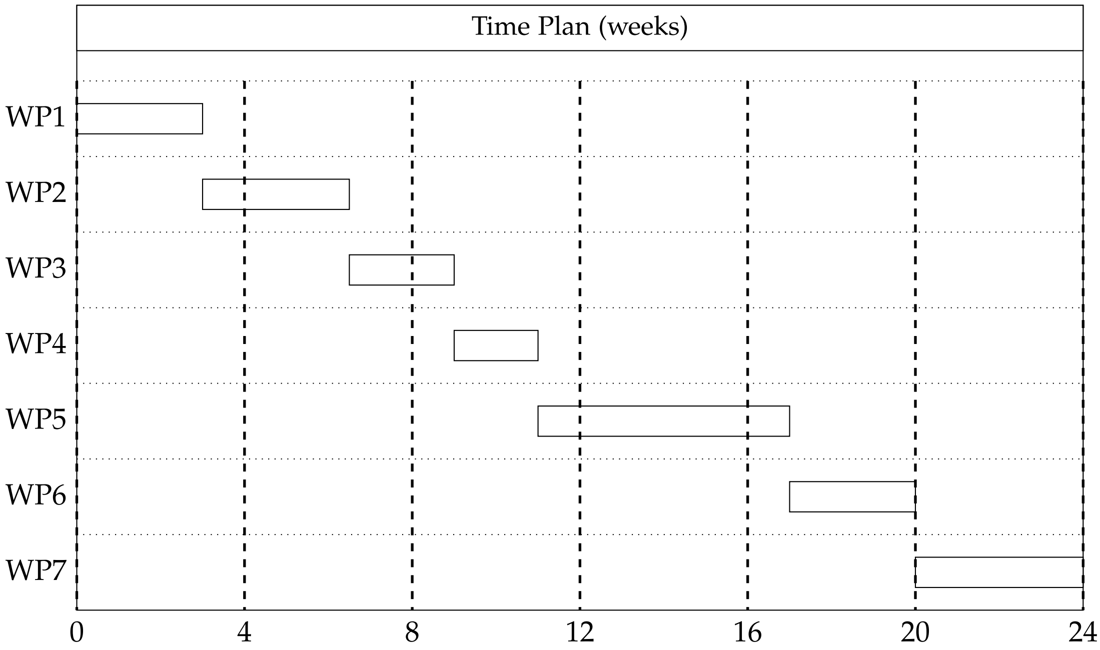
4 Necessary Software and Hardware Components

In order to fulfill the purpose of the master's thesis in the planned time, I expect to need following materials:

- Setup with several embedded systems with FreeRTOS
- Work station for machine learning before test execution
- Work station for running NeuroRobotics Platform
- SpiNNaker board for executing machine learning at runtime

5 Time Plan

The work will be organized in the work packages described below. Some work packages are divided in subpackages for a more detailed description of the work comprised in them.



1. [WP1] Initial setup of the existing independent components in the full system and familiarization with previous work and the algorithms and architectures used (3 weeks)
    a. Familiarization with previous work and other literature (1.5 weeks)
    b. Setup of the software tools to be used and familiarization with them (1.5 weeks)
2. [WP2] Setup of initial NRP scenario and embedded hardware, along with their integration (3.5 weeks)
    a. Creation of initial NRP scenario and simple application (1 week)

   b. Creation of ROS topics and services that will serve as interface with model car (0.5 weeks)
   c. Setup of embedded hardware (1 week)
   d. Integration of model car hardware with NRP scenario using the topics and services created (1 week)

3. [WP3] Development of the applications to be run on the hardware (2.5 weeks)
   a. Development of path planning application (1.5 weeks)
   b. Development of dummy applications to ensure critical system utilization (1 week)

4. [WP4] Creation of application failure scenario (2 weeks)

5. [WP5] Setup and integration of SpiNNaker board for performing the schedulability analysis and planning the task distribution (6 weeks)
   a. Setup and integration of SpiNNaker board (1.5 weeks)
   b. Implementation of the machine learning algorithm (2.5 weeks)
   c. Integration and implementation of the task distribution strategy (2 weeks)

6. [WP6] Collection and analysis of the results (3 weeks)
   a. Refinement of HIL setup to show that the system can recover from the failure scenario implemented using the task distribution strategy (1.5 weeks)
   b. Timing analysis of the results (1.5 weeks)

7. [WP7] Conclusion of the work and writing of the thesis (4 weeks)

Bibliography

[Bli19] B. Blieninger. *Online Machine Learning for improved decision making regarding the migration of automotive software components at runtime*. 2019.

[But11] G. C. Buttazzo. *Hard Real-Time Computing Systems*. 2011. DOI: 10.1007/978-1-4614-0676-1.

[Mun+17] P. Mundhenk, G. Tibba, L. Zhang, F. Reimann, D. Roy, and S. Chakraborty. "Dynamic platforms for uncertainty management in future automotive E/E architectures." In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2017, pp. 1–6. DOI: 10.1145/3061639.3072950.

[Tai19] S. Taieb. *Deep Learning Based Schedulability Analysis for Migration of Software Components at Runtime*. 2019.

[Utz19] T. Utz. *Vergleich von heuristischen und Deep Learning basierten Schedulability Analyse Verfahren für die Migration von Softwarekomponenten zur Laufzeit*. 2019.

[Vip+14] K. Vipin, S. Shreejith, S. A. Fahmy, and A. Easwaran. "Mapping Time-Critical Safety-Critical Cyber Physical Systems to Hybrid FPGAs." In: *2014 IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*. 2014, pp. 31–36. DOI: 10.1109/CPSNA.2014.14.

[Vip18] K. Vipin. "CANNoC: An open-source NoC architecture for ECU consolidation." In: *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2018, pp. 940–943. DOI: 10.1109/MWSCAS.2018.8624006.